Testing for randomness

Find out exactly how good random number generators are with a veritable barrage of tests

There are many random number generators available for you to use, but how can you tell how good they are? Just how random are the numbers that are returned from any given generator?

Before we even start attempting to answer that question, we need to go back to the beginning and ask exactly what a random number is. Donald Knuth, he of the multi-volume, unlikely-to-be-completed *Art of Computer Programming*, has possibly the best quote on the subject: "In a sense, there is no such thing as a random number. For example, is 2 a random number?"

With the joke, he does raise a serious point: when we talk about random numbers, we generally don't mean individual ones, we mean a sequence of numbers that 'look' random. Each number in the sequence should seem to bear no relation to either its preceding or its succeeding neighbours. We also don't particularly care *how* the numbers were generated; instead we use the results of the generator and require that the numbers we get exhibit randomness, as if they were produced by proverbial monkeys tossing coins.

If you ask someone to reel off a sequence of random digits, you'd probably get something that was anything but random. Humans are conditioned from an early age to think of certain digits as 'religious', 'lucky' or 'magic', and we tend to unconsciously favour them. So if you ask someone to think of a sequence of random digits, say 100 of them, they tend to respond with some digits more often than others. Also, the average person shies away from saying the same digit twice in a row, or giving sequential digits (4, 5, 6, say), both of which would be expected to crop up in a true random sequence.

Looking at the problem from the opposite direction, we humans are also extraordinarily ▸

	Two heads	One of each	Two tails
Observed results	28	51	21
Probability of event	0.25	0.5	0.25
Expected results	25	50	25

▲ **Table 1: Results for tossing two coins 100 times.**

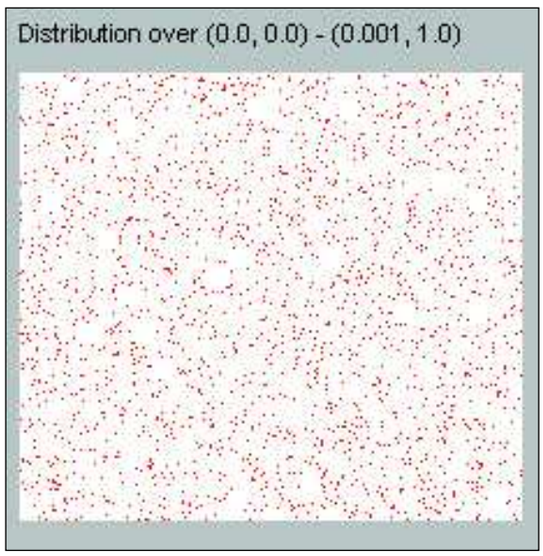
bad at determining whether a sequence is random or not. If you were presented with a sequence of digits that had three 4s in a row, would you reject it as not random? What if it had four of them in a row? Or five? Or 42? Using elementary probability, the chance of getting any given digit in any random sequence is 1/10. Hence, the probability of two 4s in a row would be 0.01, three of them 0.001, and so on. Of course, any other sequence of length n would have the same probability of appearing as n successive 4s. It's just that n 4s clumped together in a random sequence looks less than random to our eyes.

Determining randomness

So how can we decide if a random number sequence is, in fact, genuinely random? There's nothing for it but to use statistics. The first and simplest statistical test of randomness is to count all the 0s, all the 1s, all the 2s, and so on in the sequence we're given. We'd expect the count of each digit to be about a 10th of the total. So, if we had a 100-digit sequence, we'd expect to see roughly 10 of each digit.

However, although simple to calculate, this test seems a little haphazard and doesn't really tell us much. What if the counts were exactly 10 each? That sounds a little fishy to me for a purely random sequence. One would expect that in a given sequence of 100 digits some digits would be better represented than others, just by chance. You'd imagine that there would be some sloppiness in the counts, but they'd all be bunched around 10, plus or minus a few of course. But how much of a spread from exactly 10 can we allow before it all starts getting a little fishy again? Is there some more rigorous principle that we can apply here?

Time for a thought experiment. Imagine we had a pair of coins that we think are biased. How could we prove that they are? What we would do is toss the coins 100 times, say, and plot the number of times we get two heads, two tails or one of each in a table. Our table might look like Table 1. I've also added the probability and the expected number of results for each event to the table for reference.



▲ **Figure 1: A good random number plot in 2D.**

Just looking at the table, we could possibly argue that the coins are biased towards heads and away from tails, but is the difference really that significant? Let's look at the spread (the difference) of our results from the expected values. We'll square these differences to accentuate them and to get rid of any negative values. The sum of these squared differences would be a measure of how biased these coins are. Calculating this sum, I get a figure of 26 (= 32 + 12 + 42). So is that big or small, noteworthy or unimportant?

In order to answer this question, we should incorporate the probability of each event somehow. We should get a smaller term to add into our sum for 'one of each' than for 'two heads', just because the former is more likely to happen. To put it another way, the difference of 3 for 'two heads' seems much more significant than the difference of 1 for 'one of each'. So let's divide each squared difference by the expected result of that event. The new sum we calculate (which is usually known as X) is the sum of the squared spreads divided by the expected values. I get a value for X of 1.02 (= 32/25 + 12/50 + 42/25). That looks awfully scientific, but what new knowledge has it given us?

In fact, what we've just calculated is the chi-squared value for our tests ('chi' is pronounced 'ky'). We can look up this value in a standard table of the chi-squared distribution (Table 2 is an abbreviated example of the full table). The values shown in the table are selected values from the chi-squared distribution with various degrees of freedom. Without being too rigorous about it, the number of degrees of freedom is one less than the number of 'buckets' we are counting events into. In our case, we have three buckets: one for two heads, one for one of each and one for two tails, so the number of degrees of freedom for our experiment is two.

Look along the line for two degrees of freedom, and there are four values in the four columns. If we look in the one per cent column the value there (0.0201) should be read as meaning: 'The value X we calculated should be less than 0.0201 one per cent of the time'. In other words, if we repeated our experiment 100 times, only one of them (or so) would have an X value of less than 0.0201. If we found that several had a value less than 0.0201 then it would give a very strong indication that flipping the coins is not a random event and that they are biased. A similar interpretation can be made for the five per cent column.

Moving to the 95 per cent column, the value there should be read as: 'Our value X should be less than 5.99 95 per cent of the time', or, if you prefer, 'X should be greater than 5.99 only five per cent of the time'. A similar explanation also applies to the 99 per cent column.

We see that our X value falls in between the five per cent value and the 95 per cent value, and so we don't have a strong conclusion either way: we have to assume that the coins are true.

If, on the other hand, our X value were 10, we would see that this result should only occur

Tossing a coin

Remarkably, tossing a coin is not necessarily as fair as you may think. In the early '90s, Persi Diaconis showed in a set of experiments with a mechanical coin flipper that, in general, if a coin is flipped in exactly the same way (same force, same speed and so on) it will land in the same way. That is, the probability is no longer 50/50; instead, it depends on the initial state. Human flippers tend to add some randomness to this mechanical result, but it has been shown that if you train yourself to flip a coin in the same way you can alter the probability to produce a real bias, without the toss looking any different to normal. ■

Spotlight on… Plotting random points

Another way of testing whether a sequence of numbers is random or not is to plot them in a graph and then look to see if you can find patterns in the dots.

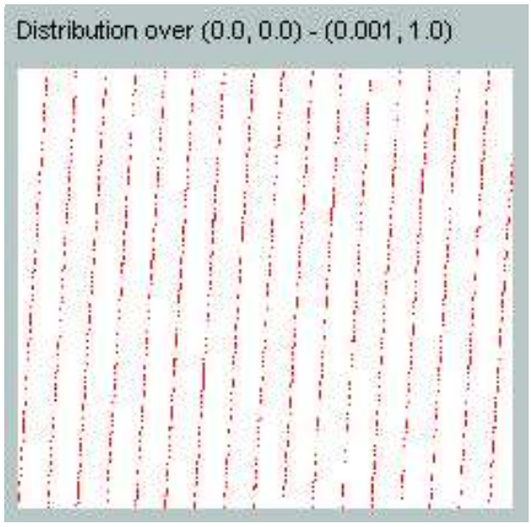
To do this you take the numbers from your generator as values between 0 and 1. You pair them as (x, y) coordinates and plot them. For a good generator, you would see something similar to Figure 1: the dots should be spread randomly across the entire graph, with no real clumping or patterns visible. (In fact, Figure 1 shows a very thin vertical slice of the graph with x varying from 0.0 to 0.001, but magnified 1,000 times horizontally.)

For a not so good generator (in this case, the so-called Minimal Standard Generator, which isn't used any more), the graph looks like Figure 2 for the same thin vertical slice, magnified. As you can see, the MSG has some very troubling attributes when the random numbers are picked off in pairs.

Generators may pass this 'pattern test' in two dimensions, but these regular effects may become visible in higher dimensions. ■

in less than one per cent of our trials (10 > 9.21, which is the 99 per cent value). And this would be a strong indication that the coins were biased. Of course, we should perform more experiments and see how our spread of X values fit into the chi-squared distribution – from an extended set we'll get a better feel for the bias of the coins. We don't want to get caught out with a rogue result, one which probability theory tells us should happen, albeit infrequently.

Generally, we take the same boundaries at either end of the range of the chi-squared distribution, for example, five per cent and 95 per cent, and then say that our experiment is significant at the five per cent level if it falls outside of these boundaries, or is not significant at the five per cent level if it falls in between them.



▲ **Figure 2: A bad random number plot in 2D.**

How many individual events should we generate? In our coin test we did 100 flips. Is this enough, or can we get away with less? The answer is unclear. Knuth states that a common rule of thumb is to make sure that the expected number of events for each bucket should be five or more (our expected numbers are 25, 50 and 25, so we're all right there), and that the more events to bucket the merrier.

Now that we have a general statistical test, let's take a look at how we can apply it to formulate four randomness tests for a sequence of numbers.

Testing the numbers

The tests all follow the same logic. We'll use random numbers between 0.0 and 1.0. Note that the ranges discussed here are all inclusive at the lower end and exclusive at the upper end. We count various events derived from these random numbers into buckets, calculate the probability associated with each bucket and, from this, work out the chi-squared value and apply the chi-squared test with the number of degrees of freedom being one less than the number of buckets. It sounds a little bit abstract but you'll see a demonstration of the idea in a moment.

The first test is the simplest, and it's called the uniformity test. This is the one we were discussing earlier. The random numbers are going to be checked to see that they 'uniformly' cover the range 0.0 to 1.0. We create 100 buckets, generate 10,000 random numbers and slot them into each bucket. Bucket 0 gets all the random numbers from 0.0 to 0.01, bucket 1 gets those from 0.01 to 0.02, and so on. The probability of a random number falling into each particular bucket is 0.01. We calculate the chi-squared value for our test and check that against the standard table, using the 99 degrees of freedom line.

The second test – the gap test – is a little more interesting. This is designed to check that you don't get runs of values in one particular range followed by runs in by another, flip-flopping between the two, even though as a whole the random numbers are evenly spread out. Define a sub-range of the range 0.0 to 1.0 – let's say 0.0 to 0.5. Now generate random numbers. For each random number, test to see whether it is in our sub-range (a 'hit') or not (a 'miss'). You'll get a sequence of hits and misses. Look at the runs of one or more misses (these are called the 'gaps' between the hits, hence the gap test). You'll get some runs of just one miss, of two misses and so on. Bucket these lengths. Let's say the probability of a hit is p (it'll be the width of the sub-range expressed as a decimal) and so the probability of a miss is $(1\text{-}p)$. We can now calculate the probability of a run of one miss $((1\text{-}p).p)$, of two misses $((1\text{-}p)2.p)$ and therefore of n misses: $((1\text{-}p)n.p)$. We can hence calculate the expected numbers for each run length. From then it's a short step to the chi-squared test. We shall use 10 buckets, hence there are nine degrees of freedom. Generally, we repeat the gap test for the first and second halves of the 0.0 to 1.0 range, and for the first, second and third thirds.

Degrees of freedom	1%	5%	95%	99%
1	0.000157	0.00393	3.84	6.63
2	0.0201	0.103	5.99	9.21
3	0.115	0.352	7.81	11.3
4	0.297	0.711	9.49	13.3
5	0.554	1.15	11.1	15.1

▲ **Table 2: Selected percentage points of the chi-squared distribution.**

The third test is known as the poker test. The random numbers are grouped into sets or 'hands' of five, and the numbers are converted into 'cards'; each 'card' actually being a digit from 0 to nine. The number of different cards in each hand is then counted (it'll be from one to five), and this result is bucketed. Because the probability of only one digit being repeated five times is so low, it's generally grouped into the 'two different digits' category. Apply the chi-squared test to the four buckets; there will be three degrees of freedom. The probability for each bucket is somewhat difficult to calculate (and involves something called Stirling numbers), so I won't attempt to explain it here.

The fourth test is the coupon collector's test. The random numbers are read one by one and converted into a 'coupon', or a number from 0 to 4. The length of the sequence required to get a complete set of the coupons the digits 0 to 4) is counted; this will obviously vary from 5 upwards. Once a full set is obtained, we start over. We bucket the lengths of these sequences and then apply the chi-squared test to the buckets. We'll use buckets for the sequence lengths from 5 to 19, and then have a composite bucket for every length after that. So, 16 buckets and hence 15 degrees of freedom. Again, like the poker test, the calculation of the probability for each bucket is somewhat mathematically intensive, so we won't present it here.

There are many other tests that can be devised along these lines. The problem, in general, is how to calculate the expected values (that is, the probability) of each event because some of them require some fairly heavy-duty mathematics. Nevertheless, these statistical tests provide the best way to determine whether a particular random number sequence is really random or not. ■

Julian M Bucknall has worked for companies ranging from TurboPower to Microsoft and is now CTO for Developer Express.
feedback@pcplus.co.uk

The Mersenne twister

Perhaps the best modern pseudo-random number generator is the Mersenne twister algorithm, which was devised in 1997 by Makoto Matsumoto and Takuji Nishimura. It's fast, has a quite astoundingly huge period (the count of random numbers generated before it repeats) of 10^6000 and passes all of Marsaglia's Diehard randomness tests. It's been implemented in the run-times for Python and Ruby, as well as applications like MATLAB. ■

More randomness tests

In 1995, George Marsaglia of Florida State University published the result of several years' work on a CD-ROM. The work was in two parts: a huge table of random numbers (4.8 billion random bits), and a battery of statistical tests coded in C. Marsaglia had produced a good dozen new tests beyond those documented by Knuth (the simpler ones we demonstrate in the main article) and the random bits on the CD-ROM passed all those tests. The table of bits was produced by the best pseudo-random number generators laced with the output from hardware devices that produce random bits using radioactive materials or electronic noise. ■